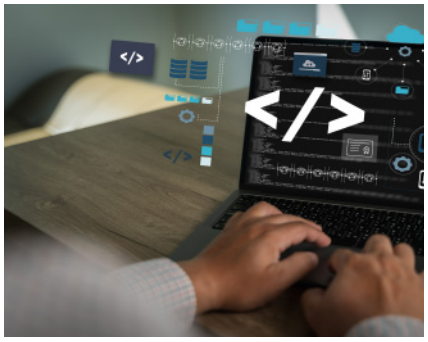
TECNOLÓGICA DEL ORIENTE
INSTITUCIÓN DE EDUCACIÓN SUPERIOR

DESARROLLO WEB

REACT: ESTADO Y PROPS

REACT - ESTADO Y PROPS: GESTIÓN DE LA INFORMACIÓN EN LA INTERFAZ



La gestión de estado y propiedades en React constituye el mecanismo fundamental a través del cual los datos fluyen a través de jerarquías de componentes, habilitando interfaces dinámicas que responden a interacciones del usuario, respuestas de API, y condiciones cambiantes de la aplicación. Las propiedades (props) sirven como el mecanismo primario para pasar datos hacia abajo desde componentes padre hacia componentes hijo, estableciendo contratos claros que definen qué información cada componente necesita para funcionar apropiadamente. El estado representa datos que pertenecen a componentes específicos y pueden cambiar a lo largo del tiempo en respuesta a acciones del usuario, solicitudes de red, u otros eventos, requiriendo consideración cuidadosa de dónde debe vivir el estado y cómo debe actualizarse para mantener consistencia y predictibilidad de la aplicación.

La gestión moderna de estado en React ha evolucionado desde setState de componentes de clase hacia enfoques basados en hooks que proporcionan control más granular y mejores oportunidades para reutilización de lógica, incluyendo useState para estado local de componentes, useReducer para transiciones de estado complejas, y useContext para compartir estado a través de árboles de componentes sin perforación de propiedades (Granados La Paz, 2023). Patrones avanzados incluyen normalización de estado para gestionar datos relacionales, actualizaciones optimistas para mejorar rendimiento percibido, e integración con bibliotecas de gestión de estado externas como Redux Toolkit para aplicaciones con requisitos de estado global complejos. La elección entre estado local, estado elevado, y estado global depende de factores como alcance de datos, frecuencia de actualización, y el número de componentes que necesitan acceso a piezas específicas de información.

En aplicaciones de comercio financiero, la gestión sofisticada de estado se vuelve crítica para manejar datos de mercado en tiempo real, información de cartera del usuario, y flujos de trabajo complejos de gestión de órdenes. Por ejemplo, un tablero de comercio debe gestionar estado local para interacciones UI como niveles de zoom de gráficos y rangos de tiempo seleccionados, estado elevado para coordinar datos entre componentes relacionados como listas de seguimiento y vistas de cartera, y estado global para autenticación de usuario, suscripciones de datos de mercado, y órdenes activas que necesitan ser accesibles a través de la aplicación. El manejo apropiado de actualizaciones de estado asíncronas, estados de error, e indicadores de carga asegura que los usuarios tengan retroalimentación clara sobre frescura de datos y estado de la aplicación durante operaciones críticas de comercio.

Ejercicio práctico. Implementar un sistema comprehensivo de gestión de estado para un tablero de medios sociales utilizando varias técnicas de gestión de estado de React.

- **Primero,** establecer gestión de estado local utilizando useState para interacciones UI como visibilidad de modales, valores de entrada de formularios, y selecciones temporales, demostrando cuándo el estado debe permanecer local a componentes específicos.

- **Segundo,** implementar patrones de estado elevado para compartir datos entre componentes hermanos, como preferencias de filtro que afectan tanto listado de publicaciones como estadísticas de barra lateral, mostrando cómo identificar cuándo el estado necesita elevarse a componentes padres comunes.
- **Tercero,** utilizar useContext y useReducer para crear gestión de estado global para autenticación de usuario, sistema de notificaciones, y preferencias de tema que necesitan ser accesibles a través de la aplicación.
- **Cuarto,** implementar actualizaciones optimistas para interacciones del usuario como gustos de publicaciones y comentarios, demostrando cómo manejar actualizaciones UI optimistas mientras gestiona conflictos potenciales cuando fallan llamadas API.

El resultado es una arquitectura robusta de gestión de estado que demuestra mejores prácticas para diferentes escalas de complejidad de estado en aplicaciones React.